UNSUPERVISED WORD ACQUISITION FROM SPEECH USING PATTERN DISCOVERY

Alex Park and James R. Glass

MIT Computer Science and Artificial Intelligence Laboratory
32 Vassar Street, Cambridge, MA 02139, USA
{malex, jrg}@csail.mit.edu

ABSTRACT

In this paper, we present an unsupervised method for automatically discovering words from speech using a combination of acoustic pattern discovery, graph clustering, and baseform searching. The algorithm we propose represents an alternative to traditional methods of speech recognition and makes use of the acoustic similarity of multiple realizations of the same words or phrases. On a set of three academic lectures on different subjects, we show that the clustering component of the algorithm is able to successfully generate word clusters that have good coverage of subject-relevant words. Moreover, we illustrate how to use the cluster nodes to retrieve the word identity of each cluster from a large baseform dictionary. Results indicate that this algorithm may prove useful for applications such as vocabulary initialization, speech summarization, or augmentation of existing recognition systems.

1. INTRODUCTION

Over the past few decades, data storage capacity, transfer speed, and computational power have all increased at extraordinary rates. In the area of speech and audio, this has led to a massive increase in the volume and variety of available audio data. Audio recordings that are now routinely archived include educational lectures, web casts, radio programs, and meetings. Although these sources have many potential uses, their utility is limited by the capacity of humans to listen to and digest the information contained therein. Unlike text, which can be easily skimmed, summarized, and searched, untranscribed audio data is tedious to browse, making it difficult to access without time-consuming data preparation.

In order to address this problem, researchers are devoting more effort to the problem of automatically annotating audio data, with the goal of increasing the utility of general audio recordings. The majority of approaches start by using automatic speech recognition (ASR) systems to generate a text transcription which is then used as the basis for further processing. This approach has some drawbacks, which we note here. First, as with all recognition systems, transcription accuracy is dependent on the amount and quality of training data used for acoustic modeling and language modeling. Next, because speech recognizers use a pre-specified lexicon, there may be many instances of words in the test data that are out of vocabulary (OOV). While the OOV rate can be reduced by increasing the vocabulary size, extraneous words in the lexicon are also undesirable. In particular, we have observed that for many educational lectures, the active vocabulary is typically very small, but includes many topic specific terms and phrases that are not frequently occurring in conversational speech [1]. Recognition experiments on these lectures show that reducing the OOV rate improves accuracy, but that including unnecessary words is detrimental to performance [2]. These experiments underscore the challenge of balancing coverage and generality against relevance and compactness when performing vocabulary selection.

In this paper, we present a conceptually simple algorithm which can analyze an audio recording and automatically discover and identify recurring patterns which are likely to be significant with respect to the content of the recording. The initial component of our algorithm is a completely unsupervised method for grouping acoustically similar patterns into clusters using pair-wise comparisons. Unlike the ASR-based approach, OOV words are not an issue, because the clustering process does not utilize a pre-specified lexicon. These clusters can potentially be used in many ways: for directly summarizing the audio stream, for initializing a lexicon which can be used to perform more exhaustive recognition, or for performing information retrieval (IR) tasks on the recording. In [3], we first introduced and demonstrated the utility of our pattern discovery method for augmenting audio IR. Here, we take the additional step of performing lexical access by using the clusters to automatically identify words in the audio stream without explicitly relying on an ASR system.

The rest of this paper is organized as follows: Sections 2, 3, and 4 are concerned with the mechanics of grouping together similar sounding speech segments using dynamic time warping (DTW) and a bottom-up clustering algorithm. While important in a practical sense, any improvements described in these sections can be seen as refinements to the ideas introduced in our previous paper [3]. The major contribution of this work is described in Sections 5, 6, and 7, where we introduce a novel method for associating words to clusters and apply the technique to a more diverse set of data.

2. SEGMENTAL DTW

Historically, DTW has been used to find the optimal global alignment between two whole word exemplars [4]. In the basic formulation, two speech waveforms are first converted into spectral vector time series, $\{\mathbf{v}_i\}_{i=1}^{M}$, $\{\mathbf{v}_j\}_{j=1}^{N}$. Next, a distance matrix D is computed by taking the pairwise distance between vectors in each utterance such that $D(i,j) = \|\mathbf{v}_i - \mathbf{v}_j\|$. Finally, dynamic programming is used to find the lowest distortion path through the distance matrix between $D(1,1)$ and $D(M,N)$. This result is useful to us because the path distortion directly measure the similarity of two utterances at the acoustic level. However, the globally optimal alignment is not suitable for our purposes because the start and

Support for this research was provided in part by the National Science Foundation under grant #IIS-0415865.

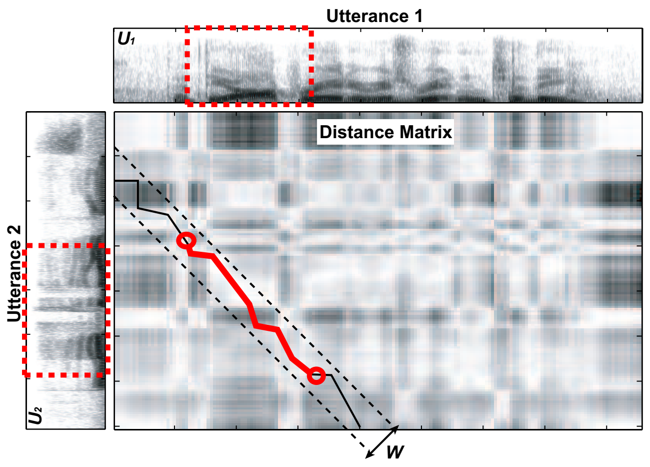

Fig. 1. The segmental DTW algorithm. As in normal DTW, the distance matrix is computed between utterances U_1 and U_2. The matrix is then cut into overlapping diagonals (only one shown here) with width W. The optimal alignment path within each diagonal band is then found using DTW and the resulting path is trimmed to the least average subsequence. Finally, the trimmed alignment paths are retained as the subsequence alignments between U_1 and U_2.

end points constrain the algorithm to match the entire utterances together, and not local alignments which correspond to matching *sub*sequences.

In order to address these limitations, we introduced a segmental variation of DTW in [3]. This algorithm is illustrated and explained in Figure 1 and is similar to the approach used for musical pattern analysis in [5] . The reasons for the constrained paths are twofold. First, via the width parameter W, they limit the amount of temporal distortion that can occur between two sub-utterances during alignment. Second, they allow for multiple alignments, as each band corresponds to another potential path with start and end points that differ from $D(1,1)$ and $D(M,N)$. The algorithm used to trim the paths finds the least average subsequence with minimum length L [6]. The minimum length criterion is used to prevent spurious matches between short segments within each utterance. After trimming, the resulting path fragments represent matching subsequences between the utterances.

In practice, we use silence detection to break the audio stream into separate utterances and then repeat the segmental DTW process for each pair of utterances. As utterances are compared against each other, the concentration of path fragments at particular locations in the audio stream will indicate higher recurrence of the pattern occuring at that location.

3. NODE EXTRACTION

The result of the segmental DTW phase is a set of alignment paths distributed throughout the audio stream. Although the alignment paths overlap common time regions, path boundaries typically do not coincide with each other and multiple intervals exist for each point in time. In [3], we showed how to extract a set of time indices from the audio stream by aggregating the inverted distortion profiles of the alignment paths to form a similarity profile over time. After smoothing the similarity profile with a triangular averaging window, we take the peaks from the resulting smoothed profile and use them to represent the multiple paths overlaying that point

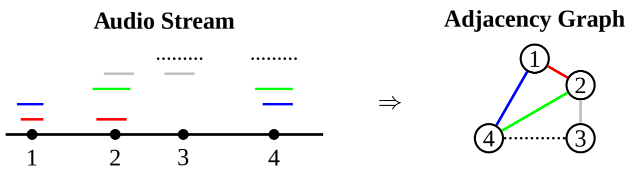

Fig. 2. Production of an adjacency graph from alignment paths and extracted nodes. The audio stream is shown as a timeline, while the alignment paths are shown as pairs of colored lines at the same height above the timeline. Node relations are captured by the graph on the right, with edge weights given by the path similarities.

in time. The extracted time indices demarcate locations that bear resemblance to other locations in the audio stream.

The reasoning behind this procedure can be understood by noting that only some portions of the audio stream will have high similarity (i.e. low distortion) to other portions. By focusing on the peaks of the aggregated similarity profile, we restrict ourselves to finding those locations that are most similar to other locations. Since every alignment path covers only a small segment, the similarity profile will fluctuate over time. This causes the audio stream to separate naturally into multiple nodes corresponding to distinct patterns that can be joined together in an adjacency graph as shown in Figure 2.

4. GRAPH CLUSTERING

In [3], we performed a similar graph conversion as described in the previous section, but used an edge weight threshold in order to separate the graph into groups of connected components. These connected components were then demonstrated to correspond strongly to relevant words and phrases from the audio. Although we relied on explicit separability to dictate the clustering results in that work, more sophisticated algorithms for automatic graph clustering have been proposed by a number of researchers in other fields [7, 8, 9].

A detailed treatment of the graph clustering problem is outside of the scope of this paper. Instead, we focus on an efficient, bottom-up algorithm proposed Newman [8]. Starting with all edges removed and each node in its own group, the algorithm merges groups together in a greedy fashion by adding edges back to the graph in the order that maximizes a modularity measure, Q, which is given by

$$Q = \sum_i (e_{ii} - a_i^2)$$

where e_{ij} is the fraction of edges in the original network that connect vertices in group i to those in group j, and $a_i = \sum_j e_{ij}$. As noted by Newman, Q is the fraction of edges that fall within groups, minus the expected value of the same quantity if edges fall at random without regard for the community structure of the graph. The value of Q ranges between 0 and 1, with higher scores indicating more favorable partitionings of the graph. The advantages of this algorithm are threefold. It easily allows us to incorporate edge weight information in the clustering process, it is extremely fast, and it includes a data-driven criterion for determining how many clusters a graph should break into.

Because our goal is to separate the graph into groups joining nodes sharing the same word(s), multiple groups containing the same word are more desirable than fewer groups containing many different words. We therefore associate a higher cost with

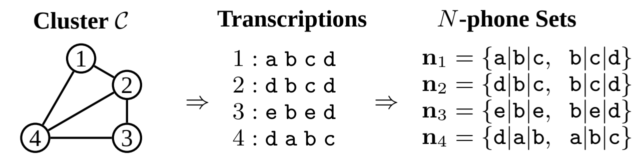
Cluster $\mathcal{C}$
Transcriptions
N-phone Sets
1 : a b c d
2 : d b c d
3 : e b e d
4 : d a b c
$\mathbf{n}_1 = \{$a|b|c, b|c|d$\}$
$\mathbf{n}_2 = \{$d|b|c, b|c|d$\}$
$\mathbf{n}_3 = \{$e|b|e, b|e|d$\}$
$\mathbf{n}_4 = \{$d|a|b, a|b|c$\}$

Fig. 3. Conversion of cluster nodes into groups of n-phones. The intervals under the nodes are phonetically transcribed, then separated into sets of n-phone sequences. In this example, $n = 3$.

the action of mistakenly joining two unlike groups than that of mistakenly leaving two like groups unmerged. This observation leads us to choose a conservative stopping point for the clustering algorithm at 80% of peak modularity.

5. CLUSTER IDENTIFICATION

The procedure we use to assign words to the clusters generated from the previous section is relatively straightforward. For a given cluster, $\mathcal{C}$, a phonetic recognizer is used to transcribe the interval underlying each node and convert it into a set of n-phones, as shown in Figure 3. Likewise, the pronunciations for all words in a large baseform dictionary (150K words, in our case), $\mathcal{W}$, are converted into sets of n-phones. This process reduces each node, $\mathbf{n}_i$, and each word, $\mathbf{w}_j$, into sets of n-phone sequences. By comparing the similarity of the words in the dictionary to the nodes in $\mathcal{C}$, the most likely candidate word common to the cluster can be found. The hypothesized cluster identity is given by

$$\mathbf{w}^* = \arg\max_{\mathbf{w}_j \in \mathcal{W}} \frac{2}{|\mathcal{C}|} \sum_{\mathbf{n}_i \in \mathcal{C}} \frac{|\mathbf{n}_i \cap \mathbf{w}_j|}{|\mathbf{n}_i| + |\mathbf{w}_j|}.$$

In this equation, we use the normalized intersection between the sets $\mathbf{n}_i$ and $\mathbf{w}_j$ as a measure of similarity and aggregate this over all nodes in the cluster for each word. We include the $|\mathbf{n}_i| + |\mathbf{w}_i|$ factor in the denominator to normalize for the size of $\mathbf{n}_i$ and $\mathbf{w}_i$, so that longer/shorter words are not favored due to their length. The factor of 2 in the numerator is included so that the overall score ranges between 0 and 1. Using this similarity score, we can easily generate an N-best list of word candidates for each cluster.

6. EXPERIMENTAL DETAILS

6.1. Speech Data

The speech data used in this paper is taken from a corpus of audio lectures collected at MIT [1]. The entire corpus consists of approximately 300 hours of lectures from a variety of academic courses and seminars. The audio was recorded using an omni-directional microphone in a classroom environment. In a previous paper, we described characteristics of this lecture data and performed recognition and information retrieval experiments [2]. Each lecture typically contains a large amount of speech (from thirty minutes to an hour) from a single person in an unchanging acoustic environment. On average, each lecture contained only 800 unique words, with high usage of subject-specific words and phrases. Each of the lectures used in our experiments was taken from courses in computer science (CS), physics, and linear algebra.

6.2. Preprocessing and Path Detection

Each lecture is segmented into continuous utterances by using a basic phone recognizer to identify regions of silence in the signal. Silent regions with duration longer than 2 seconds are removed and the portions of speech in between those silences are used as the input to the segmental DTW algorithm. In the absence of a phone recognizer, segmentation can also be performed using a speech activity detector. Each segment was then converted into a series of 14-dimension MFCC vectors extracted every 5 ms, which were whitened using principal components analysis over all vectors from all utterances. After extracting MFCC feature vectors, we compute common paths between each utterance using segmental DTW. In this phase, each utterance is compared against each other utterance. For our experiments, we set the parameters W and L to be 15 (75 ms) and 100 (500 ms), respectively.

7. RESULTS & DISCUSSION

We first consider the effectiveness of the clustering component of our algorithm. Our use of pattern discovery for finding relevant words is motivated by the assumption that relevant words and phrases in the lecture are spoken more frequently than non-important ones. We attempted to verify this by listing the top ten most frequent words from each lecture when ranked using term frequency-inverse document frequency (TFIDF). These words are shown in Table 1. Each lecture is counted as a separate document in calculating the TFIDF measure. The lists of words generated in this table appear to be very relevant to the subject matter of each lecture and indicate, qualitatively, that our assumption is valid.

An encouraging observation is that the majority of the words in Table 1 are, in fact, found as clusters in the output of our algorithm. It should be noted that for two sets of words, “square root” and “glass rod”, the words are found co-located in the same cluster. In the linear algebra lecture, the word “equation” was not found in its own cluster because the nodes were merged into a larger cluster corresponding to the word “combination” due to the acoustic similarity of the word endings. Other words that are not included, such as “LISP”, and “z”, may be due to the shorter duration of these words as spoken by the lecturers.

In Table 2, we show example clusters from each lecture ranked by size and include the results of the cluster identification algorithm. At first glance, we found that the identification procedure worked surprisingly well given that we did not use a unigram language model to bias the baseform dictionary prior to search. We chose to show individual clusters rather than simply compute overall identification accuracy because we found the types of errors made by the algorithm particularly enlightening.

The type a errors (indicated with superscript^a) originate with the clustering component of our algorithm. The clusters erroneously combine acoustically similar, but lexically dissimilar nodes consisting of function word sequences, and word components. Examples include {“this is”, “misses”, “which is”} and {“would be”, “b”, “see”}. The lexical disagreement for these clusters can be seen in their purity scores, which measures what fraction of the nodes contain the most common reference word(s) for the cluster. An important observation we can make, however, is that the scores for the hypothesized words are very low, indicating that we can use the identification score as a metric for rejecting these types of clusters.

Type b and c errors can be attributed to limitations of the iden-

CompSci	Algebra	Physics
root*	column*	charge*
LISP	plane*	force*
define*	x*	balloon*
square*	matrix*	electron*
procedure*	equation	glass*
language	combination*	rod*
primitive*	z	electricity*
x*	solution	atom*
computer*	y*	positive*

Table 1. Ten most frequent words for each lecture ranked by term frequency, inverse document frequency (TFIDF) measure. Each lecture is counted as a separate document. The starred* words occur as found clusters in the output of our algorithm.

tification procedure. Multi-word phrase clusters induce type b errors, where the best matching single word in the dictionary only partially covers the reference phone sequence. In the case of "right hand side" and "square root", the algorithm finds one of the constituent words, but for "times ten to", the best matching word is "tent", which occurs phonetically, but not lexically, in the phrase.

Type c errors are characterized by single-word clusters with relatively high purity. Upon examining the node phonetic transcriptions, we concluded that the errors for these cases is due to the inability of the phonological rules to account for the surface realization of the word. Conspicuous examples are all present in the CS lecture, where the lecturer consistently omits the "b" in "combination" and omits both schwas in "parentheses". In the future, we may be able to mitigate these errors by using more powerful phonological rules or by adopting a more flexible search phase. It is interesting to note that almost all of the type c errors occurred in the CS lecture, which suggests that the occurrence of these errors is highly dependent on speaking style.

8. SUMMARY & FUTURE WORK

In this paper, we have described a method for identifying words from speech without using conventional speech recognition techniques. The algorithm we propose relies on the recurrence of words in the audio stream and makes use of the acoustic similarity of multiple realizations of the same word. We observed that our algorithm can successfully find many clusters corresponding to important words relevant to each lecture. Using our cluster identification method, we can also, in many cases, identify the correct word corresponding to these clusters.

In the future, we plan to improve upon the results we have presented by iterating the word discovery process and by using phone lattices instead of the top phone transcription during the cluster identification stage. We can also incorporate lexical knowledge to help identify clusters by using the word N-best lists for each cluster to find the set of words that maximizes some joint probability of all words occurring in a single document. Even without the cluster identification component, we believe that our approach has many potential uses. In many tasks involving the organization of large amounts of audio data, the core idea of pattern discovery may be more suitable than a traditional speech recognizer because it is completely language independent and requires no training data. The unsupervised nature of the algorithm also makes it useful for improving our understanding of how to learn directly from speech.

	Size	Common Word(s)	Purity	Hypothesis	Score
(a)	72	square_root^b	1.00	square	0.23
	40	procedure	0.97	procedure	0.34
	21	combination^c	1.00	commination	0.43
	19	computer	1.00	computer	0.63
	17	primitive	0.94	primitive	0.12
	14	definition	1.00	definition	0.29
	12	parentheses^c	1.00	prentice	0.29
	12	product^c	0.83	prada	0.65
	10	operator	0.80	operator	0.37
	10	and	1.00	and	0.29
(b)	73	combination	0.52	combination	0.26
	46	be^a	0.24	woodby	0.03
	45	column	1.00	column	0.85
	42	and^a	0.64	manthe	0.04
	32	minus	0.94	minus	0.45
	30	matrix	0.97	matrix	0.69
	27	is^a	0.56	get	0.08
	22	right_hand_side^b	0.95	righthand	0.46
	14	picture	1.00	picture	0.88
	11	one_and^b	1.00	wanna	0.26
(c)	21	is^a	0.24	paced	0.05
	18	charge	1.00	charge	0.76
	17	positively	0.76	positively	0.48
	16	electricity	1.00	electricity	0.71
	9	forces	0.89	forces	0.72
	9	positive	0.89	positive	0.94
	7	gravitational	1.00	invitational	0.61
	6	times_ten_to^b	1.00	tent	0.57
	6	distance	0.83	distance	0.51
	5	gravity	1.00	gravity	0.44

Table 2. Ten largest clusters for lectures in (a) computer science, (b) linear algebra, and (c) physics. From left to right, the columns list cluster size, the underlying common reference word(s) for the cluster, the cluster purity (fraction of nodes containing the common reference word(s)), the top cluster identity hypothesis ($\mathbf{w}^*$), and the hypothesis score.

9. REFERENCES

[1] J. Glass, T. J. Hazen, L. Hetherington, and C. Wang, "Analysis and processing of lecture audio data: Preliminary investigations," in *Proc. HLT-NAACL 2004 Workshop on Interdisciplinary Approaches to Speech Indexing and Retrieval*, Boston, May 2004, pp. 9–12.

[2] A. Park, T. J. Hazen, and J. Glass, "Automatic processing of audio lectures for information retrieval: Vocabulary selection and language modeling," in *Proc. ICASSP*, Philadelphia, 2005, pp. I–497–450.

[3] A. Park and J. Glass, "Towards unsupervised pattern discovery in speech," in *Proc. IEEE Workshop on Automatic Speech Recognition and Understanding*, San Juan, Puerto Rico, 2005.

[4] L. Rabiner and B.H. Juang, *Fundamentals of Speech Recognition*, Prentice Hall, 1993.

[5] R. Dannenberg and Ning Hu, "Pattern discovery techniques for music audio," in *International Conference on Music Information Retrieval*, Paris, France, 2002, pp. 63–70.

[6] Y-L. Lin, T. Jiang, and K-M. Chao, "Efficient algorithms for locating the length-constrained heaviest segments with applications to biomolecular sequence analysis," *J. Computer and System Sciences*, vol. 65, no. 3, pp. 570–586, January 2002.

[7] S. White and P. Smyth, "A spectral clustering approach to finding communities in graphs," in *SIAM International Conference on Data Mining*, Newport Beach, CA, 2005.

[8] M. E. J. Newman, "Fast algorithm for detecting community structure in networks," *Phys. Rev. E*, vol. 69, pp. 066133, 2004.

[9] S. van Dongen, *Graph Clustering by Flow Simulation*, Ph.D. thesis, University of Utrecht, 2000.